CLAIMS

1. (**CURRENTLY AMENDED**) A computer-implemented apparatus having a processor for handling thread requests in a disparate computer environment, wherein the disparate computer environment arises because the threads requesting services are preemptively scheduled entities ~~operate asynchronously with respect to each other~~ whereas service agents servicing the requests are cooperatively scheduled entities, ~~operate synchronously with respect to each other,~~ wherein the threads requesting services are preemptively scheduled entities because processing of the threads requesting services can be temporarily interrupted, said apparatus comprising:

~~a first lock configured to operate as~~ a dispatch lock ~~for allowing only one requesting thread into a dispatch section at a time~~ that is acquired by the requesting threads to allow a thread to gain access to a dispatch section one at a time;

~~a second lock configured to operate as~~ a service pool lock for synchronizing ~~the requesting thread that is in the dispatch section with a service agent~~ one of the requesting threads, that has gained access to the dispatch section, with one of the service agents;

wherein the dispatch lock and the service pool lock are nested relative to each other such that the service pool lock is acquired while holding the dispatch lock in order to synchronize the service agent with the requesting thread that has gained access to the dispatch section;

wherein, after the requesting thread releases the ~~first~~ dispatch lock and ~~second~~ the service pool lock, the service agent handles the request of the requesting thread.

2. (**CANCELLED**)

3. (**ORIGINAL**) The apparatus of claim **1**, wherein the requesting thread prepares parameters that are to be passed to the synchronized service agent.

4. (**ORIGINAL**) The apparatus of claim **1**, further comprising:

a dispatch module which passes the parameters to the synchronized service agent.

5. (**ORIGINAL**) The apparatus of claim **4**, wherein the dispatch module operates in the requesting thread's context.

6. (**ORIGINAL**) The apparatus of claim **4**, wherein the dispatch module selects a service agent that is free from a pool of services.

7. (**ORIGINAL**) The apparatus of claim **6**, wherein if a free service agent is not available from the pool, then the dispatch module requests and awaits the creation of another service agent that can perform the request of the thread.

8. (**ORIGINAL**) The apparatus of claim **7** further comprising:

a spawner that creates another service agent based upon the request from the dispatch module.

9. (**ORIGINAL**) The apparatus of claim **6**, wherein if a free service agent is not available from the pool, then the dispatch module waits on an extant service agent to complete its assignment, wherein the extant service agent is used to service the request of the thread.

10. (**ORIGINAL**) The apparatus of claim **6**, wherein when the service agent completes a request, notification is provided to the waiting requesting thread, and reenters the pool of free service agents in order to await another request from a requesting thread.

11. (**CURRENTLY AMENDED**) The apparatus of claim **1**, wherein the ~~first~~ dispatch lock involves a synchronization point for dispatching the service request.

12. (**CURRENTLY AMENDED**) The apparatus of claim **11**, wherein the ~~first~~ dispatch lock does not involve the requesting thread awaiting completion of a service agent that is handling the request of the thread.

13. (**CURRENTLY AMENDED**) The apparatus of claim **1**, wherein the service agent is task-based in that services are ~~synchronous~~cooperatively scheduled.

14. (**ORIGINAL**) The apparatus of claim **13**, wherein the task-based service agent operates in a single-threaded environment.

15. (**ORIGINAL**) The apparatus of claim **13**, wherein the task-based service agent operates in a cooperative multi-tasking environment, wherein only one task-based service agent can execute at a time.

16. (**CURRENTLY AMENDED**) The apparatus of claim **15**, wherein a first pool of services includes task-based service agents, wherein a second pool of services includes thread-based service agents, wherein the ~~first~~ dispatch lock and ~~second~~ service pool lock[[s]] are used in accessing the first and second pools of service agents.

17. (**ORIGINAL**) The apparatus of claim **1**, wherein utilization of the requesting threads constitutes a technological advance over the use of the service agents.

18. (**PREVIOUSLY PRESENTED**) The apparatus of claim **17**, wherein the service agents constitute a legacy system which becomes substantially compatible with the requesting threads through use of the first and second locks;

wherein the legacy system includes task-based code that becomes compatible with the requesting thread through utilization of the first and second locks.

19. (**CURRENTLY AMENDED**) A computer-implemented apparatus for servicing thread requests, comprising:

a ~~first~~ dispatch lock that is acquired by a requesting thread that allows only one requesting thread into a dispatch section at a time;

a ~~second~~ service pool lock that synchronizes the requesting thread that is in the dispatch section with a service agent;

wherein, after the requesting thread releases the ~~first~~ dispatch and ~~second~~ service pool lock, the service agent handles the request of the requesting thread.

20. (**ORIGINAL**) The apparatus of claim 1, wherein the requesting thread waits for the results from the service agent handling the request.

21. (**CURRENTLY AMENDED**) A computer-implemented apparatus for handling thread requests in a disparate computer environment, wherein the disparate computer environment arises because the threads requesting services operate in a multi-threaded processing environment whereas the agents servicing the request operate in a single-threaded processing environment, comprising:

a ~~first~~ dispatch lock that is acquired by a requesting thread that allows only one requesting thread into a dispatch section at a time;

a ~~second~~ service pool lock that synchronizes the requesting thread that is in the dispatch section with a service agent;

wherein, after the requesting thread releases the ~~first~~ dispatch and ~~second~~ service pool lock, the service agent handles the request of the requesting thread.

22. (**CURRENTLY AMENDED**) A computer-implemented apparatus for utilizing legacy computer code with multi-threaded code, wherein the multi-threaded code generates service requests, wherein the legacy computer code handles a service request in a single-threaded processing environment, comprising:

a ~~first~~ dispatch lock that is acquired by a requesting thread allows ~~a~~ one requesting thread into a dispatch section at a time;

a ~~second~~ service pool lock that synchronizes the requesting thread that is in the dispatch section with a service agent;

wherein the synchronized service agent handles the request of the thread.

23. (NEW) A processor-implemented method for handling thread requests in a disparate computer environment, wherein the disparate computer environment arises because the threads requesting services are preemptively scheduled entities whereas service agents servicing the requests are cooperatively scheduled entities, wherein the threads requesting services are preemptively scheduled entities because processing of the threads requesting services can be temporarily interrupted, said method comprising:

acquiring a dispatch lock by the requesting threads to allow a thread to gain access to a dispatch section;

using a service pool lock to synchronize one of the requesting threads, that has gained access to the dispatch section, with a service agent;

wherein the dispatch lock and the service pool lock are nested relative to each other such that the service pool lock is acquired while holding the dispatch lock in order to synchronize the service agent with the requesting thread that has gained access to the dispatch section;

wherein, after the requesting thread releases the dispatch lock and the service pool lock, the service agent handles the request of the requesting thread.

24. (NEW) The method of claim **23**, wherein the requesting thread prepares parameters that are to be passed to the synchronized service agent.

25. (NEW) The method of claim **23**, further comprising:

passing parameters from a dispatch module to the synchronized service agent.

26. (NEW) The method of claim **25**, wherein the dispatch module operates in the requesting thread's context.

27. (NEW) The method of claim **25**, wherein the dispatch module selects a service agent that is free from a pool of services.

28. **(NEW)** The method of claim **27**, wherein if a free service agent is not available from the pool, then the dispatch module requests and awaits the creation of another service agent that can perform the request of the thread.

29. (NEW) The method of claim **28**, further comprising:

creating another service agent based upon the request from the dispatch module.

30. (NEW) The method of claim **25**, wherein if a free service agent is not available from the pool, then the dispatch module waits on an extant service agent to complete its assignment, wherein the extant service agent is used to service the request of the thread.

31. (NEW) The method of claim **25**, wherein when the service agent completes a request, notification is provided to the waiting requesting thread, and reenters the pool of free service agents in order to await another request from a requesting thread.

32. (NEW) The method of claim **23**, wherein the dispatch lock involves a synchronization point for dispatching the service request.

33. (NEW) The method of claim 32, wherein the dispatch lock does not involve the requesting thread awaiting completion of a service agent that is handling the request of the thread.

34. (NEW) The method of claim 23, wherein the service agent is task-based in that services are cooperatively scheduled.

35. (NEW) The method of claim 34, wherein the task-based service agent operates in a single-threaded environment.

36. (NEW) The method of claim 34, wherein the task-based service agent operates in a cooperative multi-tasking environment, wherein only one task-based service agent can execute at a time.

37. (NEW) The method of claim 36, wherein a first pool of services includes task-based service agents, wherein a second pool of services includes thread-based service agents, wherein the dispatch lock and the service pool lock are used in accessing the first and second pools of service agents.

38. (NEW) The method of claim 23, wherein utilization of the requesting threads constitutes a technological advance over the use of the service agents.

39. (**NEW**) The method of claim **38**, wherein the service agents constitute a legacy system which becomes substantially compatible with the requesting threads through use of the first and second locks;

wherein the legacy system includes task-based code that becomes compatible with the requesting thread through utilization of the first and second locks.